DIPLOMADO INGENIERIA DE CALIDAD
DE SOFTWARE COMERCIAL (3ra Edición)
CARRERA DE INGENIERÍA INFORMÁTICA

SEXTO MÓDULO
AUTOMATIZACIÓN DE PRUEBAS

Grupo: **SoftSign**

Integrantes:

- Alvarez Cayo Elvis
- Gutierrez Orellana Kevin
- Delgadillo Fernandez Pablo Enrique
- Navia Luna Edwin Efrain
- Trujillo Delgadillo Neida Jessica
- Cantarran Villarroel Noelia

Docente: Enrique Rivera

Cochabamba - Bolivia

1. Introducción

El presente documento detalla el plan de pruebas diseñado para la aplicación estilo Trello, cuyo propósito es garantizar la calidad funcional del sistema mediante la evaluación de sus principales módulos. El enfoque de las pruebas se centra principalmente en la interfaz de usuario (UI), validando la usabilidad, consistencia y cumplimiento de los flujos de trabajo, complementado con comprobaciones básicas en el backend a través de la API REST, con el fin de asegurar la correcta gestión y persistencia de datos.

El alcance de este plan incluye la revisión de las funciones esenciales de la plataforma: creación y administración de tableros, columnas y tarjetas, así como las extensiones asociadas a estas últimas (etiquetas, fechas, checklist, adjuntos, ubicación y campos personalizados). Con ello, se busca detectar de manera temprana posibles desviaciones respecto a los requisitos establecidos y aportar información precisa para la mejora continua del sistema.

2. Descripción del producto

La aplicación Trello es una plataforma de gestión de proyectos y tareas basada en tableros.
Su funcionamiento principal se centra en:

- Crear tableros para organizar proyectos.
- Crear columnas (listas) para representar estados o categorías.
- Crear tarjetas (cards) que contienen la información de cada tarea.
- Extender funcionalidades de las tarjetas con etiquetas, fechas, checklist, adjuntos, ubicación y campos personalizados.
- Interacción vía UI Web y API REST para la administración de datos.

3. Objetivos

3.1. Objetivo General

Verificar la calidad funcional de la aplicación Trello a través de pruebas en UI y API REST, asegurando que las funcionalidades principales de tableros, columnas y tarjetas se ejecuten correctamente.

3.2. Objetivos específicos.

- Validar el correcto inicio de sesión (Login).

- Comprobar la creación y gestión de tableros, columnas y tarjetas básicas.
- Verificar la extensión de funcionalidades de las tarjetas
    - ❖ Etiquetas.
    - ❖ Fechas.
    - ❖ Checklist.
    - ❖ Adjuntos.
    - ❖ Campos personalizados.
- Evaluar la coherencia entre los datos mostrados en la UI y los devueltos por la API REST.
- Identificar posibles fallas funcionales y de integración.
- Documentar los resultados y evidencias para trazabilidad.

4. Funcionalidades

4.1. Login

- Inicio de sesión con credenciales válidas.
- Manejo de errores con credenciales inválidas.
- Uso de tokens de sesión en la API.

4.2. Módulo Boards (Tableros)

- Crear, editar, eliminar tableros.
- Validar permisos de acceso.

4.3 Módulo Lists (Listas dentro de un tablero)

- Crear, editar, mover y eliminar listas.
- Reordenar listas con drag & drop.

4.4. Módulo Cards (Tarjetas dentro de listas)

- Crear, editar, mover y eliminar tarjetas.
- Asignar usuarios, etiquetas y fechas límite.
- Validar notificaciones y actualizaciones en tiempo real.

4.5. API Trello (principales endpoints)

- Boards: /1/boards (GET, POST, PUT, DELETE)
- Lists: /1/lists (GET, POST, PUT)
- Cards: /1/cards (GET, POST, PUT, DELETE)

5. Límites y alcances

5.1. Alcance

- Validación de UI Web: interacción, usabilidad y validaciones visuales.
- Validación de API REST: endpoints de creación, actualización, consulta y eliminación.
- Pruebas funcionales, exploratorias y de integración básica.

5.2. Fuera de alcance

- Pruebas de carga, estrés y escalabilidad.
- Integraciones con servicios externos (Google Drive, Slack, etc.).
- Pruebas de seguridad avanzadas (penetration testing).

6. Herramientas

- Postman / Insomnia → pruebas de API REST.
- Playwright / Node js → pruebas automatizadas de UI.
- Pytest + Requests – pruebas automatizadas API.
- Allure – reportes de resultados.
- Qase – gestión de casos de prueba.
- Taiga – gestión de sprints y backlog.
- GitHub Actions – CI/CD para pruebas automatizadas.

7. Tipos de pruebas

- Exploratory Testing (UI + API): descubrir comportamientos no esperados.

  Pruebas exploratorias realizadas sin un guión de prueba predefinido, combinando el diseño y la ejecución de las pruebas de forma simultánea. Permiten descubrir comportamientos, defectos o inconsistencias en la aplicación a través de la exploración dinámica, basándose en la experiencia, el conocimiento del dominio y la intuición del tester.

- Smoke Testing: login, carga de tableros, creación de tarjeta.

  Validación básica y rápida de los endpoints clave del sistema, como login y recursos principales, para asegurar que el sistema responde correctamente antes de ejecutar pruebas más complejas.

- Functional Testing: CRUD de boards/lists/cards, drag & drop.
  Verificación de que cada endpoint cumple con el comportamiento esperado ante entradas válidas. Se aseguran respuestas correctas, estructuras de datos adecuadas y cumplimiento de la lógica de negocio.

- Negative Testing: inputs inválidos, permisos incorrectos
  Evaluación del sistema frente a entradas incorrectas, formatos inválidos, parámetros faltantes o rutas erróneas. Permite identificar fallos de validación o manejo inadecuado de errores.

  End-to-End (E2E): flujo completo login → crear tablero → listas → tarjetas → compartir con usuario

- Validación de flujos completos de extremo a extremo para asegurar la correcta integración y funcionamiento de los componentes del sistema en conjunto.

8. Metodología

8.1. Scrum y kanban

Se aplicará una combinación de Scrum y Kanban para la gestión ágil del proyecto:

- Scrum: planificación de sprints, reuniones diarias, retrospectivas.
- Kanban: visualización y priorización de tareas en tableros.

Los artefactos principales serán:

- Product Backlog – funcionalidades y pruebas planificadas.
- Sprint Backlog – casos priorizados por iteración.
- Test Reports (Allure/Qase) – resultados de ejecución.

9. Recursos

9.1. Equipo QA:

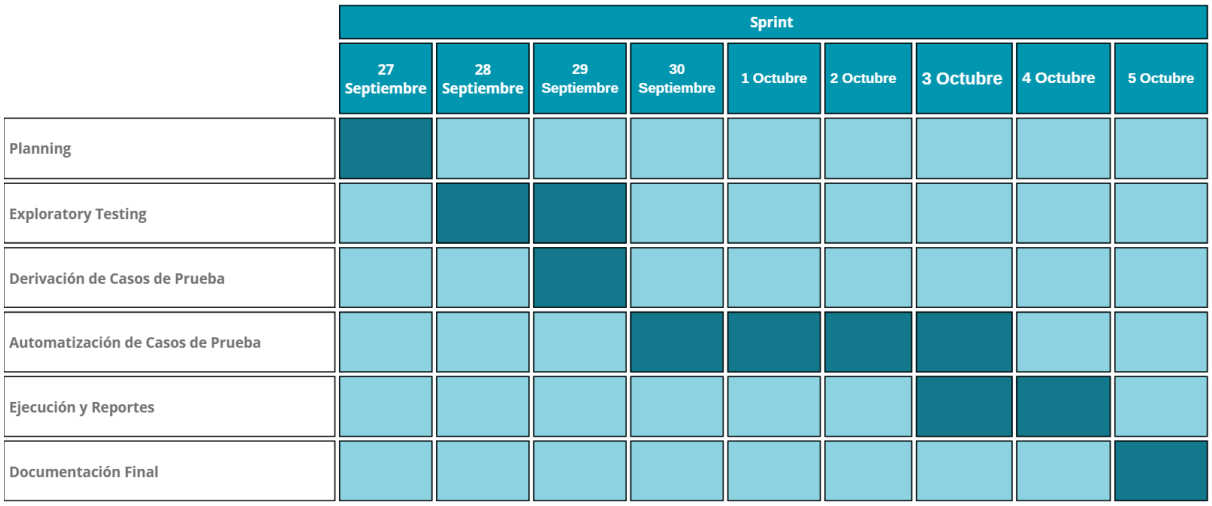
- Base (Login, Tablero, Columnas, Tarjeta básica): Pablo, Elvis.
- Boards (Front y Back): Kevin.
- Tarjetas – Etiquetas: Elvis.
- Tarjetas – Fechas: Edwin.
- Tarjetas – Checklist: Jessica.
- Tarjetas – Adjuntos: Kevin.
- Tarjetas – Campos personalizados: Noelia.

9.2. Recursos técnicos:

- Navegador Chrome
- Entorno local con Node.js.
- Acceso a endpoints REST.
- Herramientas de prueba (Postman, Playwright, VS Code).

10. Cronograma

10.1. Sprint 1



Rol	Nombre
QA Lead	Kevin Gutierrez Orellana
QA Team	Elvis Alvarez Cayo Pablo Enrique Delgadillo

	Fernandez Kevin Gutierrez Orellana Edwin Efraín Navia Luna Neida Jessica Trujillo Delgadillo Noelia Cantarran Villarroel

11. Test Cases

RESUMEN GENERAL:
- Total de Casos de Prueba: 13 casos específicos + Login dinámico
- Cobertura: Gestión de tableros, tarjetas, dashboard, login y flujo E2E
- Herramienta: Playwright con TypeScript
- Datos de prueba: Generados dinámicamente

MÓDULO: Boards

TC001 - CREACIÓN EXITOSA DE TABLERO CON FUNCIONALIDAD BÁSICA
Título: Verificar la creación exitosa de tablero con listas por defecto y funcionalidad básica de tarjetas
Descripción: Validar que un usuario puede crear un tablero nuevo, generar las listas por defecto (To Do, In Progress, Done) y crear una tarjeta básica
Prioridad: ALTA
Pasos:
1. Acceder al dashboard
2. Crear un nuevo tablero con nombre generado
3. Crear listas (To Do, In Progress, Done)
4. Crear una tarjeta básica en la lista To Do
5. Validar que la tarjeta sea visible
Resultado Esperado: El tablero se crea exitosamente con las listas por defecto y la tarjeta es visible
Archivo: tests/ui/board.spec.ts

TC002 - CREACIÓN DE MÚLTIPLES TARJETAS EN DIFERENTES LISTAS
Título: Verificar la creación de múltiples tarjetas en diferentes listas del tablero (To Do, In Progress, Done)

Descripción: Validar que se pueden crear múltiples tarjetas distribuidas en las diferentes listas del tablero
Prioridad: ALTA
Pasos:
1. Crear un tablero con listas
2. Crear una tarjeta en la lista "To Do"
3. Crear una tarjeta en la lista "In Progress"
4. Crear una tarjeta en la lista "Done"
5. Validar que todas las tarjetas sean visibles
Resultado Esperado: Las tres tarjetas se crean exitosamente en sus respectivas listas y son visibles
Archivo: tests/ui/board.spec.ts

TC003 - MODIFICACIÓN DE NOMBRE DE TABLERO
Título: Verificar la funcionalidad de modificación de nombre de tablero y validación
Descripción: Validar que un usuario puede modificar el nombre de un tablero existente
Prioridad: MEDIA
Pasos:
1. Crear un tablero
2. Editar el nombre del tablero
3. Validar que el cambio de nombre se refleje correctamente
Resultado Esperado: El nombre del tablero se actualiza correctamente
Archivo: tests/ui/board.spec.ts

TC004 - ACCESIBILIDAD DEL MENÚ INBOX
Título: Verificar la accesibilidad del menú inbox y visualización adecuada desde la interfaz del tablero
Descripción: Validar que el menú inbox es accesible y se muestra correctamente
Prioridad: BAJA
Pasos:
1. Crear un tablero
2. Abrir el menú Inbox
3. Validar que el menú Inbox se muestre correctamente
Resultado Esperado: El menú Inbox es accesible y se visualiza adecuadamente
Archivo: tests/ui/board.spec.ts

TC005 - ACCESIBILIDAD DEL MENÚ PLANNER

Título: Verificar la accesibilidad del menú planner y visualización adecuada desde la interfaz del tablero
Descripción: Validar que el menú planner es accesible y se muestra correctamente
Prioridad: BAJA
Pasos:
1. Crear un tablero
2. Abrir el menú Planner
3. Validar que el menú Planner se muestre correctamente
Resultado Esperado: El menú Planner es accesible y se visualiza adecuadamente
Archivo: tests/ui/board.spec.ts

MÓDULO: Cards

TC006 - CARGA DE ARCHIVO DE IMAGEN A TARJETA
Título: Verificar la funcionalidad de carga exitosa de archivo de imagen a tarjeta con validación
Descripción: Validar que se puede subir un archivo de imagen como adjunto a una tarjeta
Prioridad: ALTA
Pasos:
1. Crear configuración básica con listas y tarjeta
2. Crear una tarjeta de prueba
3. Agregar archivo de imagen a la tarjeta
4. Validar que el archivo se haya subido correctamente
Resultado Esperado: El archivo de imagen se adjunta exitosamente a la tarjeta
Archivo: tests/ui/card.spec.ts

TC007 - CARGA DE ARCHIVO JSON A TARJETA
Título: Verificar la funcionalidad de carga exitosa de archivo JSON a tarjeta con validación
Descripción: Validar que se puede subir un archivo JSON como adjunto a una tarjeta
Prioridad: MEDIA
Pasos:
1. Crear configuración básica con listas y tarjeta
2. Crear una tarjeta de prueba

3. Agregar archivo JSON a la tarjeta
4. Validar que el archivo se haya subido correctamente
Resultado Esperado: El archivo JSON se adjunta exitosamente a la tarjeta
Archivo: tests/ui/card.spec.ts

TC008 - ASIGNACIÓN DE FECHA DE VENCIMIENTO
Título: Verificar la funcionalidad de asignación de fecha de vencimiento con fecha de inicio y configuración de recordatorio
Descripción: Validar que se puede asignar una fecha de vencimiento a una tarjeta con recordatorios
Prioridad: ALTA
Pasos:
1. Crear configuración básica con listas y tarjeta
2. Crear una tarjeta de prueba
3. Agregar fecha específica a la tarjeta
4. Validar que la fecha se haya asignado correctamente
Resultado Esperado: La fecha de vencimiento se asigna correctamente con recordatorios configurados
Archivo: tests/ui/card.spec.ts

TC009 - CREACIÓN DE CHECKLIST CON MÚLTIPLES ELEMENTOS
Título: Verificar la creación de checklist con múltiples elementos y funcionalidad de estado de completado
Descripción: Validar que se puede crear una checklist con múltiples elementos y marcar su estado
Prioridad: ALTA
Pasos:
1. Crear configuración básica con listas y tarjeta
2. Crear una tarjeta de prueba
3. Agregar checklist a la tarjeta
4. Validar que el checklist se haya creado correctamente
Resultado Esperado: El checklist se crea con múltiples elementos y permite gestión de estado
Archivo: tests/ui/card.spec.ts

TC010 - ASIGNACIÓN DE ETIQUETAS A TARJETAS
Título: Verificar la funcionalidad de asignación de etiquetas e identificación visual en tarjetas
Descripción: Validar que se pueden asignar etiquetas a tarjetas para identificación visual
Prioridad: MEDIA

Pasos:
1. Crear configuración básica con listas y tarjeta
2. Crear una tarjeta de prueba
3. Agregar etiquetas a la tarjeta
4. Validar que las etiquetas se hayan asignado correctamente
Resultado Esperado: Las etiquetas se asignan correctamente y son visualmente identificables
Archivo: tests/ui/card.spec.ts

MÓDULO: DASHBOARD

TC011 - CREACIÓN DE TABLERO DESDE DASHBOARD
Título: Verificar la creación exitosa de tablero desde dashboard y validación de visibilidad
Descripción: Validar que un tablero creado desde el dashboard sea visible y accesible
Prioridad: ALTA
Pasos:
1. Acceder al dashboard
2. Crear un nuevo tablero
3. Regresar al dashboard
4. Validar que el tablero sea visible en el dashboard
Resultado Esperado: El tablero se crea exitosamente y es visible en el dashboard
Archivo: tests/ui/dashboard.spec.ts

TC012 - CIERRE DE TABLERO Y ACTUALIZACIÓN DE VISIBILIDAD
Título: Verificar la funcionalidad de cierre de tablero y actualización de visibilidad en dashboard
Descripción: Validar que un tablero cerrado no sea visible en el dashboard principal
Prioridad: MEDIA
Pasos:
1. Crear un tablero
2. Cerrar el tablero
3. Regresar al dashboard
4. Validar que el tablero no sea visible en el dashboard
Resultado Esperado: El tablero cerrado no aparece en la vista principal del dashboard
Archivo: tests/ui/dashboard.spec.ts

TC013 - ELIMINACIÓN COMPLETA DE TABLERO
Título: Verificar el proceso completo de eliminación de tablero y remoción del dashboard
Descripción: Validar que un tablero eliminado se remueve completamente del dashboard
Prioridad: ALTA
Pasos:
1. Crear un tablero
2. Eliminar el tablero completamente
3. Validar que el tablero no sea visible en el dashboard
Resultado Esperado: El tablero se elimina completamente y no aparece en el dashboard
Archivo: tests/ui/dashboard.spec.ts

MÓDULO: AUTENTICACIÓN

CASOS DE LOGIN DINÁMICOS
Título: Validación de funcionalidad de login con múltiples usuarios
Descripción: Ejecuta pruebas de login tanto para usuarios válidos como inválidos de forma dinámica
Prioridad: ALTA

Usuarios de Prueba:
- Usuarios inválidos: Cargados desde data/users.json (invalid, invalid2, invalid3, invalid4)
- Usuario válido: Cargado desde variables de entorno (.env)

Funcionalidades Validadas:
- Login exitoso para usuarios válidos
- Fallo de login para usuarios inválidos
- Manejo de autenticación de dos factores (MFA)
- Redirección correcta después del login
- Manejo de errores de autenticación

Resultado Esperado:
- Usuarios válidos acceden exitosamente
- Usuarios inválidos fallan apropiadamente
- MFA funciona correctamente cuando es requerido
Archivo: tests/ui/login.spec.ts

MÓDULO: PRUEBA END-TO-END (E2E)

TC050 - FLUJO COMPLETO END-TO-END DE TRELLO
Título: Verificar el flujo completo end-to-end: creación de tablero, configuración de listas, gestión de tarjetas con adjuntos, fechas, checklists y transiciones de flujo de trabajo
Descripción: Validar el flujo completo de trabajo en Trello desde la creación hasta el archivado
Prioridad: ALTA
Duración: 3 minutos (180 segundos)

Flujo Completo:
1. Crear tablero
2. Configurar listas (To Do, In Progress, Done)
3. Crear tarjeta inicial
4. Agregar fecha a la tarjeta
5. Agregar checklist a la tarjeta
6. Agregar archivo a la tarjeta
7. Validar configuración completa de la tarjeta
8. Verificar que las características se mantengan después de las transiciones
9. Mover tarjeta a "In Progress"
10. Mover tarjeta a "Done"
11. Archivar tarjeta
12. Validar que la tarjeta no sea visible

Resultado Esperado: Todo el flujo de trabajo se ejecuta exitosamente sin errores, incluyendo:
- Creación exitosa de tablero y listas
- Gestión completa de tarjetas con todas las funcionalidades
- Transiciones correctas entre listas
- Archivado exitoso de tarjetas
Archivo: tests/ui/E2E.spec.ts

RESUMEN DE COBERTURA
Módulos Cubiertos:
- Autenticación (Login/MFA)
- Gestión de Tableros (CRUD)

- Gestión de Tarjetas Básicas
- Funcionalidades Avanzadas de Tarjetas
- Gestión de Dashboard
- Flujo End-to-End Completo
- Navegación y Menús

Funcionalidades Principales:

- Creación y gestión de tableros
- Creación y gestión de tarjetas
- Adjuntos de archivos (imagen/JSON)
- Fechas de vencimiento y recordatorios
- Checklists y seguimiento de progreso
- Etiquetas y categorización
- Campos personalizados
- Movimiento de tarjetas entre listas
- Archivado y eliminación
- Navegación entre vistas

Tipos de Validación:

- Validación de visibilidad de elementos
- Validación de funcionalidad
- Validación de flujo de trabajo
- Validación de persistencia de datos
- Validación de estados de aplicación

TEST CASES LIST

Test Cases - SoftSign				
Módulos	**Sub-Módulos**	**Casos de Prueba**		
		ID	**TÍTULO**	**Prioridad**
UI		**TR-72**	User > Boards > UI > Creation > Verificar la creación exitosa de tablero con listas por defecto y funcionalidad básica de tarjetas	Alta
		TR-83	User > Boards > UI > Multiple Cards > Verificar la creación de múltiples tarjetas en diferentes listas del tablero (To Do, In Progress, Done)	Alta
		TR-26	User > Boards > UI > Modification > Verificar la funcionalidad de modificación de nombre de tablero y validación	Media
		TR-98	User > Boards > UI > Navigation > Verificar la accesibilidad del menú inbox y visualización adecuada desde la interfaz del tablero	Baja
		TR-131	User > Boards > UI > Navigation > Verificar la accesibilidad del menú planner y visualización adecuada desde la interfaz del tablero	Baja
		TR-122	User > Cards > UI > FileUpload > Verificar la funcionalidad de carga exitosa de archivo de imagen a tarjeta con validación	Alta
		TR-65	User > Cards > UI > FileUpload > Verificar la funcionalidad de carga exitosa de archivo JSON a tarjeta con validación	Media
		TR-140	User > Cards > UI > DueDate > Verificar la funcionalidad de asignación de fecha de vencimiento con fecha de inicio y configuración de recordatorio	Alta
		TR-109	User > Cards > UI > Checklist > Verificar la creación de checklist con múltiples elementos y funcionalidad de estado de completado	Alta
		TR-126	User > Cards > UI > Labels > Verificar la funcionalidad de asignación de etiquetas e identificación visual en tarjetas	Media
		TR-83	User > Dashboard > UI > BoardCreation > Verificar la creación exitosa de tablero desde dashboard y validación de visibilidad	Alta

		TR-89	User > Dashboard > UI > BoardClosure > Verificar la funcionalidad de cierre de tablero y actualización de visibilidad en dashboard	Media
		TR-81	User > Dashboard > UI > BoardDeletion > Verificar el proceso completo de eliminación de tablero y remoción del dashboard	Alta
		TR-45	User > Authentication > UI > Login > Validación de funcionalidad de login con múltiples usuarios	Alta
		TR-79	User > Workflow > UI > 2E2 > Verificar el flujo completo end-to-end: creación de tablero, configuración de listas, gestión de tarjetas con adjuntos, fechas, checklists y transiciones de flujo de trabajo	Alta
API	**Attachment**	**TR-67**	User > Cards >Attachment >Adjuntar imagen aleatoria a la tarjeta creada en el setup	Alta
		TR-139	User > Cards > Attachment >Adjuntar varias imágenes diferentes a la misma tarjeta	Alta
		TR-128	User > Cards >Attachment > Adjuntar imágenes con diferentes extensiones	Media
		TR-101	User > Cards >Attachment > Adjuntar imagen con URL inválida	Alta
		TR-58	User > Cards > Attachment >Adjuntar imagen a una tarjeta inexistente	Alta
		TR-91	User > Cards >Attachment > Adjuntar imagen con nombre vacío	Media
		TR-19	User > Cards > Attachment >Obtener un adjunto específico de una tarjeta	Alta
		TR-55	User > Cards > Attachment >Obtener los adjuntos de una tarjeta inexistente	Media
		TR-04	User > Cards > Attachment >Obtener un adjunto inexistente en una tarjeta válida	Media
		TR-26	User > Cards > Attachment >Eliminar un adjunto específico de una tarjeta	Alta
		TR-32	User > Cards > Attachment >Intentar eliminar un adjunto inexistente	Media
		TR-55	User > Cards >Attachment > Eliminar el mismo adjunto dos veces	Media
		TR-89	User > Cards >Attachment > Intentar eliminar un adjunto sin autenticación	Alta
	Checklist	**TR-106**	User > Cards >checklist> Crear checklist "Checklist 1" en card nueva	Media
		TR-127	User > Cards >checklist> Marcar item de "Checklist 2" como completado en card neva	Media
		TR-21	User > Cards > checklist > Eliminar checklist "Checklist 3" en card	Media

		TR-100	User > Cards > checklist > Crear checklost "Checklist 4" con item adicional	Media
		TR-42	User > Cards > checklist>Completar item de "Checklist 5"	Media
		TR-123	User > Cards >ui>checklist>Crear checklist "Checklist 6"	Media
		TR-01	User > Cards >ui>checklist>Completar item de "Checklist 7"	Media
		TR-31	User > Cards >ui>checklist>Eliminar checklist "Checklist8"	Media
		TR-128	User > Cards >ui>checklistTDD>Checklist Tests - Trello UI (Single Session)	Media
		TR-133	User > Cards >Api>checklistt>Agregar item con nombre valido	Media
		TR-121	User > Cards >Api>checklistt>Agregar item con checklist id invalido	Media
		TR-107	User > Cards >Api>checklistt>Agregar item sin nombre	Media
		TR-55	User > Cards >Api>checklistt>Agregar con tipo de dato incorrecto	Media
		TR-98	User > Cards >Api>checklistt>Agregar item a checlist eliminado	Alta
		TR-60	User > Cards >Api>checklistt>Agregaar item con nombre extremadamente largo	Alta
		TR-04	User > Cards >Api>checklistt>Crear checklist "Checlist valido"	Media
		TR-65	User > Cards >Api>checklistt>Crear checklist con nombre vacio	Alto
		TR-47	User > Cards >Api>checklistt>Crearchecklist con nombre muy largo	Media
		TR-123	User > Cards >Api>checklistt>Crear checklist con caracteres especiales	Media
		TR-46	User > Cards >Api>checklistt>Crear checklist con espacio en blanco	Meida
		TR-141	User > Cards >Api>checklistt>Crear checklist con onombre numerico	Media
		TR-26	User > Cards >Api>checklistt>Actualizar checklist con nombre valido	Media
		TR-77	User > Cards >Api>checklistt>Actualizar checklist con nombre vacio	Media
		TR-67	User > Cards >Api>checklistt>Actualizar checklist con nombre muy largo	Alta
		TR-71	User > Cards >Api>checklistt>Actualizar checklist con caracteresespeciales	Media
		TR-24	User > Cards >Api>checklistt>Actualizar checklist con espacio en blanco	Media

		TR-06	User > Cards >Api>checklistt>Actualizar checklist con nombre numerico	Media
		TR-134	User > Cards >Api>checklistt>Crear checklisst sin idcard(fallido)	Alta
	Fecha	**TR-133**	User > Cards > Api > fechas > Actualizar fechas y recordatorio correctamente en una card existente	Alta
		TR-51	User > Cards > Api > fechas > Rechazar actualización cuando la fecha de vencimiento es anterior a la de inicio	Alta
		TR-96	User > Cards > Api > fechas > Validar error al enviar dueReminder con formato no numérico	Media
		TR-47	User > Cards > Api > fechas > Verificar persistencia de fechas y recordatorio después del procesamiento interno	Alta
		TR-34	User > Cards > Api > fechas > Devolver error 404 al intentar actualizar una card inexistente	Media
	Custom Field	**TR-92**	User > Cards > API> Custom Fields > Crear un campo personalizado tipo “list” con opciones válidas	Alta
		TR-16	User > Cards > API> Custom Fields > Crear un campo personalizado con un nombre largo	Media
		TR-31	User > Cards > API> Custom Fields > Crear un campo personalizado con caracteres especiales	Media
		TR-132	User > Cards > API> Custom Fields > Crear un campo personalizado con emoji en el nombre	Baja
		TR-36	User > Cards > API> Custom Fields > Crear un campo personalizado con un tipo no soportado	Alta
		TR-138	User > Cards > API> Custom Fields > Crear un campo personalizado sin especificar el campo “type”	Alta
		TR-124	User > Cards > API> Custom Fields > Crear un campo personalizado sin el campo “idModel”	Alta
		TR-17	User > Cards > API> Custom Fields > Crear un campo personalizado tipo “text”	Alta
		TR-145	User > Cards > API> Custom Fields > Crear un campo personalizado tipo “checkbox”	Alta
		TR-133	User > Cards > API> Custom Fields > Crear un campo personalizado duplicado	Media
		TR-17	User > Cards > API> Custom Fields > Obtener un campo personalizado existente	Alta
		TR-07	User > Cards > API> Custom Fields > Obtener un campo personalizado inexistente	Alta

		TR-120	User > Cards > API> Custom Fields > Obtener un campo personalizado con ID vacío	Media
		TR-52	User > Cards > API> Custom Fields > Obtener un campo personalizado con formato de ID inválido	Media
		TR-90	User > Cards > API> Custom Fields > Intentar obtener un campo personalizado eliminado	Alta
		TR-101	User > Cards > API> Custom Fields > Obtener un campo personalizado y validar que el tipo sea “list”	Media
		TR-35	User > Cards > API> Custom Fields > Obtener un campo personalizado sin permisos válidos	Alta
		TR-99	User > Cards > API> Custom Fields > Verificar que el campo personalizado pertenece al tablero correcto	Media
		TR-38	User > Cards > API> Custom Fields > Eliminar un campo personalizado existente	Alta
		TR-05	User > Cards > API> Custom Fields > Intentar eliminar un campo personalizado inexistente	Alta
		TR-16	User > Cards > API> Custom Fields > Intentar eliminar un campo personalizado sin especificar un ID	Media
		TR-102	User > Cards > API> Custom Fields > Intentar eliminar un campo personalizado con un ID vacío	Media
		TR-57	User > Cards > API> Custom Fields > Agregar una opción válida a un campo personalizado tipo “list”	Alta
		TR-08	User > Cards > API> Custom Fields > Agregar una opción sin el campo “value.text”	Alta
		TR-93	User > Cards > API> Custom Fields > Agregar una opción con tipo de dato incorrecto en “value.text”	Alta
		TR-19	User > Cards > API> Custom Fields > una opción con un nombre extremadamente largo	Media
		TR-73	User > Cards > API> Custom Fields > Intentar agregar una opción a un campo personalizado eliminado	Alta
		TR-111	User > Cards > API> Custom Fields > Actualizar el nombre de un campo personalizado existente	Alta
		TR-37	User > Cards > API> Custom Fields > Actualizar un campo personalizado con un nombre largo	Media

		TR-126	User > Cards > API> Custom Fields > Actualizar un campo personalizado con caracteres especiales	Media
		TR-03	User > Cards > API> Custom Fields > Actualizar un campo personalizado con nombre vacío	Alta
		TR-15	User > Cards > API> Custom Fields > Actualizar un campo personalizado con nombre numérico	Baja
		TR-80	User > Cards > API> Custom Fields > Actualizar un campo personalizado con emoji en el nombre	Baja
		TR-72	User > Cards > API> Custom Fields > Intentar actualizar un campo personalizado inexistente	Alta
		TR-59	User > Cards > API> Custom Fields > Intentar actualizar un campo personalizado con ID vacío	Alta
		TR-49	User > Cards > API> Custom Fields > Actualizar el tipo de un campo personalizado (de “list” a “text”)	Alta
	Label	**TR-116**	User > Card > Label > Eliminar una Label	Alta
		TR-118	User > Card > Label > Eliminar una Label inexistente	Alta
		TR-102	User > Card > Label > Eliminar una Label ya eliminada	Media
		TR-40	User > Card > Label > Eliminar una Label sin enviar ID	Media
		TR-88	User > Card > Label > Eliminar una Label con ID inválido	Media
		TR-13	User > Card > Label > Obtener Label existente	Alta
		TR-73	User > Card > Label > Obtener Label inexistente	Alta
		TR-48	User > Card > Label > Obtener Label sin ID	Media
		TR-22	User > Card > Label > Obtener Labels de un Board	Alta
		TR-141	User > Card > Label > Obtener Labels de un Board inexistente	Media
		TR-70	User > Card > Label > Crear Label completo	Alta
		TR-109	User > Card > Label > Crear Label sin nombre	Media
		TR-129	User > Card > Label > Crear Label sin color	Media
		TR-73	User > Card > Label > Crear Label con color inválido	Media

		TR-124	User > Card > Label > Crear Label sin idBoard	Alta
		TR-137	User > Card > Label > Crear Label con idBoard inválido	Alta
		TR-113	User > Card > Label > Crear Label sin payload	Alta
		TR-85	User > Card > Label > Modificar el nombre de una Label	Alta
		TR-123	User > Card > Label > Modificar el color de una Label	Media
		TR-60	User > Card > Label > Modificar el nombre y color de una Label	Media
		TR-70	User > Card > Label > Modificar una Label con datos inválidos	Alta
		TR-125	User > Card > Label > Modificar una Label que no existe	Baja
		TR-22	User > Card > Label > Modificar una Label sin enviar datos	Baja